CHAPTER 10 Encryption

Keeping secrets by disguising them, hiding them, or making them indecipherable to others is an ancient practice. It evolved into the modern practice of cryptography—the science of secret writing, or the study of obscuring data using algorithms and secret keys.

A Brief History of Encryption

Once upon a time, keeping data secret was not hard. Hundreds of years ago, when few people were literate, the use of written language alone often sufficed to keep information from becoming general knowledge. To keep secrets then, you simply had to write them down, keep them hidden from those few people who could read, and prevent others from learning how to read. Deciphering the meaning of a document is difficult if it is written in a language you do not know.

History tells us that important secrets were kept by writing them down and hiding them from literate people. Persian border guards in the fourth century B.C. let blank wax writing tablets pass, but the tablets hid a message warning Greece of an impending attack. The message was simply covered by a thin layer of fresh wax. Scribes also tattooed messages on the shaved heads of messengers. When their hair grew back in, the messengers could travel incognito through enemy lands. When they arrived at their destination, their heads were shaved and the knowledge was revealed. But the fate of nations could not rely for very long on such obfuscation, which relied entirely on these tricks to keep the writing secret. It did not take very long for ancient military leaders to create and use more sophisticated techniques. Ever since then, encryption has been a process of using increasingly complex tactics to stay one step ahead of those who want your secret data.

When hiding meanings in ordinary written language and hiding the message itself became passé, the idea of hiding the meaning became the rule. From what we know of early civilizations, they loved a good puzzle. What greater puzzle than to obscure a message using a series of steps? Unless you knew the steps taken—the algorithm used to produce the *cipher* (the name for the disguised message)—you couldn't untangle or decipher it without a great deal of difficulty. However, the challenge of an unsolvable puzzle was irresistible to the best minds (which also remains true to this day). Eventually each code was broken, revealing each secret. Soon every side had its makers of codes and its code breakers.

Early Codes

Early code attempts used *transposition*. They simply rearranged the order of the letters in a message. Of course, this rearrangement had to follow some order, or the recipient would not be able to restore the message. The use of the scytale by the Spartans in the fifth century B.C. is the earliest record of a pattern being used for a transposition code. The scytale was a rod around which a strip of paper was wrapped. The message was written down the side of the rod, and when it was unwound, the message was unreadable. If the messenger was caught, the message was safe. If he arrived safely, the message was wound around an identical rod and read.

Other early attempts at cryptography (the science of data protection via encryption) used *substitution*. A substitution algorithm simply replaces each character in a message with another character. Caesar's cipher is an example of a substitution algorithm. To create these messages, you list the alphabet across a page and agree with the recipient on the starting letter—suppose you agree to start with the fourth letter of the alphabet, *D*. Starting with this letter, you write down a new alphabet under the old, so it looks like this:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Substitution code book

To write a coded message, simply substitute the letter in the second row every time its corresponding letter in the first row would be used in the message. The message "The administrator password is password" becomes "Wkh dgplqlvwudwru sdvvzrug lv sdvvzrug." To decipher the message, of course, you simply match the letters in the coded message to the second row and substitute the letters from the first.

These codes are interesting, but they are quite easy to break, as Mary, Queen of Scots, learned the hard way in the 16th century. Mary plotted to overthrow Queen Elizabeth of England, but her plans were found and decoded, and she was beheaded.

The use of such codes, in which knowledge of the algorithm is all that keeps the message safe, has long been known to be poor practice. Sooner or later, someone will deduce the algorithm, and all is lost. Monoalphabetic algorithms (those using a single alphabet), like the previous code, are easily broken by using the mathematics of frequency analysis. This science relies on the fact that some letters occur more often in written language than others. If you have a large enough sample of the secret code, you can apply the knowledge of these frequencies to eventually break the code. Frequency analysis is an example of cryptanalysis (the analysis of cryptographic algorithms).

Eventually, of course, variations of substitution algorithms appeared. These algorithms used multiple alphabets and could not be cracked by simple frequency analysis. One of these, the Vigenère Square, used 26 copies of the alphabet and a key word to determine which unique substitution was to be used on each letter of the message. This complex, polyalphabetic algorithm, developed by the 16th century French diplomat, Blaise de Vigenère, was not broken for 300 years. One of the reasons for its success was the infinite variety of keys—the *keyspace* that could be used. The key itself, a word or even a random combination of letters, could be of varied length, and any possible combination of characters could be used. In general, the larger the keyspace, the harder a code is to crack.

The first one-time pads were actual collections of paper—pads that had a unique key written on each page. Each correspondent possessed a duplicate of the pad, and each message used a new key from the next sheet in the pad. After its one-time use, the key was thrown away. This technique was successfully used during World War I. The key was often used in combination with a Vigenère Square. Since the key changed for each message, the impact of a deduced key only resulted in the current message being lost.

More Modern Codes

The modern stream and block ciphers used today are sophisticated encryption algorithms that run on high-speed computers, but their origins were in simple physical devices used during colonial times. An example of such an early device is the *cipher disk*. The cipher disk was actually composed of two disks, each with the alphabet inscribed around its edge. Since the diameter of the disks varied, the manner in which the alphabets lined up differed from set to set. To further complicate the matter, an offset, or starting point, was chosen. Only the possessor of a duplicate cipher disk set with knowledge of the offset could produce the same "stream" of characters.

In 1918, the German Enigma machine used the same principle but included 15,000 possible initial settings. Even the possession of the machine was no guarantee of success in breaking the code, as you had to know the setting used at the start. Imagine a series of rotators and shifters that change their location as they are used. Input the letter *D*, and after a bit, the letter *F* is output. Put in another *D* and you may get a *G* or a *U*. While this encoding may seem arbitrary, it can be reproduced if you have an identical machine and if you configure it exactly the same way. This machine was used extensively during World War II, but its code was broken with the use of mathematics, statistics, and computational ability. Early versions of the machine were actually produced for commercial purposes, and long before Hitler came to power, the code was broken by the brilliant Polish mathematician Marian Rejewski. Another version of the machine was used during World War II. Alan Turing and the British cryptographic staff at Bletchley Park used an Enigma machine provided by the Poles and again broke the codes.

Other encryption devices were produced and used during this same time period. The U.S. government's "Big Machine" looks like an early typewriter on steroids. This machine, officially known as the Sigaba, is the only known encryption machine whose code was not broken during World War II. Other machines are the Typex, designed for secure communications between the British and the Americans, the American Tunney and Sturgeon machines, which were capable of both encrypting and transmitting, and the Japanese Purple and Jade machines.

These machines, or parts of them, can be seen at the National Security Association Museum.

Symmetric-Key Cryptography

Every encryption technology from the past is still used today, in one form or another. Modern steganography hides messages in web graphics files and in the static that accompanies radio messages. School children and journal writers compose messages using simple substitution algorithms, and sophisticated one-time pads are reproduced in software and hardware tokens.

Likewise, *stream ciphers* are often produced in code today, a modern example being RC4. Programmatic stream ciphers use a key to produce a key table, and then a byte of the key table and a byte of plaintext (text that is not encrypted) are XORed. The key table is remixed and a new byte of the table is XORed with the next byte of the plaintext message. When the entire message has been thus encrypted to produce ciphertext, it is delivered. XOR, or exclusive OR, is a logic statement by which ones and zeros can be added. Since XOR is reversible, if the original key and the algorithm for producing the table are known, the ciphertext can be decrypted.

While a stream cipher works on one character at a time, *block ciphers* work on a block of many bits at a time. A nonlinear Boolean function is used. Unlike stream ciphers, early block ciphers did not vary the key, which made the results easier to break because encrypting the same combinations of letters resulted in the same ciphertext. Frequency analysis could effectively be used to break the code. Later block ciphers incorporated additional functions against the ciphertext to obscure any repetitive data. DES, once the encryption standard of the U.S. government, is a block cipher that uses 16 rounds of activity against a 64-bit block of data, with each round adding a layer of complexity. This algorithm is well known, but without the secret key (which is 40 or 56 bits in length), it is difficult to decrypt. In fact, DES was once considered so secure that it was forecast that it would take a million years before it could be broken. These days, most computers can break DES in just a few hours or less. This is a good example of overestimation based on computing power at a particular point in time. As computing power continues to increase and continues to become cheaper, in keeping with Moore's law, encryption method after encryption method will fall.

Triple DES was the next enhancement to DES. It concatenates DES ciphertext with itself and uses up to three keys, hence the name. The Advanced Encryption Standard (AES) has now replaced DES and Triple DES as the new U.S. federal government standard. AES (which is actually the Rijndael algorithm) is a cipher block algorithm that uses a 128-bit, 192-bit, or 256-bit block size and a key size of 128 bits. Other examples of block ciphers are RSA's RC2 and RC5, Entrust's CAST, and Counterpane Systems' Blowfish. Unlike DES, they can use variable-sized, large keys. For these algorithms, a larger key size results in stronger encryption. Cryptographers will continue to recommend ever-increasing key sizes for these encryption techniques, as computing power catches up to the complexity of the algorithms.

The U.S. National Institute of Standards and Technology (NIST) has published rules regarding the use of encryption within U.S. government agencies, in Special Publication 800-131A. In recognition of the limited lifespan of encryption techniques as computers grow ever more powerful, this publication specifies end-of-life dates for various cryptosystems. In this publication, Triple DES has two key lengths, known as two-key Triple DES and three-key Triple DES. Two-key Triple DES has been assessed at a security strength of 80 bits, whereas three-key Triple DES is assessed at a security strength of 112 bits. Skipjack, a block cipher developed by the U.S. National Security Agency (NSA), is assessed at a security strength of 80 bits. AES has three approved key lengths: 128, 192, and 256 bits. AES-128 is assessed at a security strength of 128 bits, AES 192 at a security strength of 192 bits, and AES-256 at a security strength of 256 bits.

The NIST transition schedule for encryption algorithms is provided in Table 10-1.

Algorithm	Use
Two-key Triple DES Encryption	Acceptable through 2010 Restricted use from 2011 through 2015 Disallowed after 2015
Two-key Triple DES Decryption	Acceptable through 2010 Legacy use after 2010
Three-key Triple DES Encryption and Decryption	Currently acceptable
SKIPJACK Encryption	Acceptable through 2010
SKIPJACK Decryption	Acceptable through 2010 Legacy use after 2010
AES-128 Encryption and Decryption	Currently acceptable
AES-192 Encryption and Decryption	Currently acceptable
AES-256 Encryption and Decryption	Currently acceptable

Table 10-1 Encryption Transitions Specified in NIST SP800-131A, Demonstrating Limited Lifespan for Cryptosystems

Key Exchange

The previously described cryptographic algorithms have at their heart the use of a single, secret key. This key is used to encrypt the data, and the same key, or a copy of it, is used to decrypt the data. This key may be used to produce other keys, but the principle is the same. These single-key, symmetric algorithms work fine as long as the key can somehow be shared between the parties that wish to use it. In the past, this has often been done by the out-of-bounds means, such as using a courier or a personal visit, or some other method that did not involve the as yet to be established communication. Over time, the needs of cryptography spread, and with this came an increasing need to frequently change keys to prevent discovery or to lessen the impact of a compromised key. It was not always possible for people to meet, or to scale the out-of-bounds method. The problem becomes significantly more difficult when you want to apply the use of cryptography to thousands of machine-generated communications.

A way to solve this problem was first proposed by Whitfield Diffie and Martin Hellman. The Diffie-Hellman key agreement protocol uses two sets of mathematically related keys and a complex mathematical equation that takes advantage of this relationship. If each of two computers calculates its own set of related keys (neither set being related to the other) and shares one of the keys with the other computer, they each can independently calculate a third secret key that will be the same on each computer. This secret key can be used independently to generate a number of symmetric encryption keys that the two computers can use to encrypt data traveling from one to the other.

Public Key Cryptography

Another method for exchanging a session key is to use *public key cryptography*. This algorithm is asymmetric—it uses a set of related keys. If one key is used to encrypt the message, the other is used to decrypt it, and vice versa. This means that if each party holds one of the keys, a session key can be securely exchanged. In the typical arrangement, each party has their own set of these asymmetric keys. One of the key pairs is known as the *private key* and the other as the *public key*. Public keys are exchanged and private keys are kept secret. Even if a public key becomes, well, public, it does not compromise the system. It's meant to be shared openly.

In addition to its use for key exchange, public key cryptography is used to create digital signatures. These algorithms traditionally use very large keys, and while you could use public key cryptography to encrypt whole messages or blocks of data on a disk, the process is remarkably slow compared to symmetric-key cryptography.

Bigger is better, but maybe not forever. Many security mechanisms, both on the Internet and off, rely on the security behind public key algorithms, such as RSA's algorithms. Public key cryptography relies, in part, on the inability of large numbers to be factored. That's why Microsoft released an update for its operating systems in August 2012 to block the use of any RSA keys less than 1024 bits in length. Computing power had become sufficient to easily factor the numbers in smaller key sizes.

Key Exchange

Public/private key pairs can be used to exchange session keys. To do so, each party that needs to exchange keys generates a key pair. The public keys are either exchanged among the parties or kept in a database. The private keys are kept secret. When it is necessary to exchange a key, one party can encrypt it using the public key of the other. The encrypted key is then transmitted to the other party. Since only the intended recipient holds the private key that is related to the public key used to encrypt the session key, only that party can decrypt the session key. The confidentiality of the session key is assured, and it can then be used to encrypt communications between the two parties.

The steps are outlined here and are illustrated in Figure 10-1. Operations 1 and 2 can take place at the same time, as can operations 3 and 4.

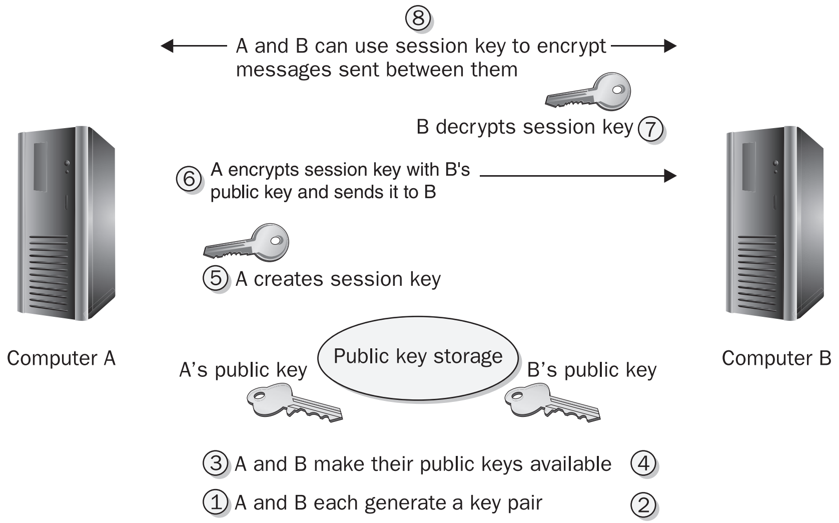
⑧
A and B can use session key to encrypt messages sent between them
B decrypts session key ⑦
⑥ A encrypts session key with B's public key and sends it to B
⑤ A creates session key
Computer A
Computer B
A's public key
Public key storage
B's public key
③ A and B make their public keys available ④
① A and B each generate a key pair ②

Figure 10-1 Using public key cryptography for key exchange

Public Key Infrastructure

Public Key Infrastructure (PKI) has become one of the most prevalent forms of encryption in modern electronic transactions. An associated key pair is bound to a security principal (user or computer) by a certificate. The certificate also makes the security principal's public key available, while the related private key is kept hidden.

A certificate authority (CA) issues, catalogs, renews, and revokes certificates under the management of a policy and administrative control. There is no need either to purchase third-party products to do so, or to purchase individual certificates from public CAs. (There may, however, be additional reasons to do that, such as to obtain an SSL certificate for a public e-commerce site.) Newer versions of Windows Server have continued to provide CA services and to add functionality. For example, Windows Server 2003 Enterprise CAs offer version 2 certificate templates and Windows Server 2008 Enterprise CAs offer version 3 certificate templates to allow for further control and customization.

Structure and Function

Multiple CAs can be arranged in a hierarchy for security, redundancy, and geographical and functional diversity. CAs can issue various types of templates. A user or computer must have a template designed and approved for a specific use in order to participate in a specific function such as encrypting files or using a smart card or enrolling other users.

CA Hierarchy

In a hierarchy, one root CA provides CA certificates for another level of CAs. While there are many hierarchical designs that can be arranged, the classic, best practice design is displayed in Figure 10-2. In this design, the root CA is kept offline and produces CA certificates only for the next, intermediary level of CAs. These CAs are integrated with AD and kept online. They issue certificates for a third level: the issuing CAs that actually issue certificates for end use such as EFS or smart cards. Issuing CAs do not issue CA certificates.

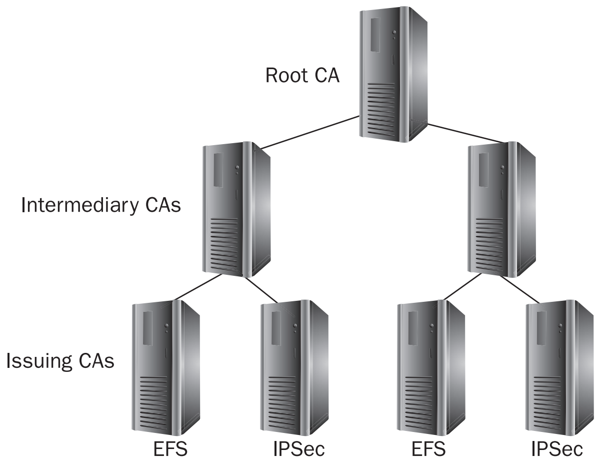

Figure 10-2 A classic Windows CA hierarchy

Certificate Templates and Enrollment

CAs integrated with Active Directory, called Enterprise CAs, issue many different types of certificates, based on built-in certificate templates. Enrollment can be automatic, manual with automatic issuance, or manual and approved by a CA Administrator. Permissions set on the templates further determine which groups of Windows users and computers can actually obtain a certificate. Windows Server 2003 introduced version 2 certificate templates, which can be customized and add features such as auto enrollment and even key archival. Key archival allows the private key associated with the certificate to be stored in a central database. This is important for recovery. Encrypted files, for example, cannot be decrypted without the private key associated with the public key used to protect the file encryption key. By archiving the EFS private keys, an organization ensures the availability of the data, even if the original keys are destroyed or damaged.

Windows Server 2008 R2 introduced version 3 templates, which add support for the newer Microsoft Crypto-API, giving administrators the ability to produce certificates using the more advanced and secure elliptic curve cryptography (ECC) cryptography service providers (CSPs).

NOTE So, you're thinking, what would prevent a malicious administrator from retrieving the private key and reading the private files? The answer lies in the design of the system and in the proper application of controls such as enforcing role separation. A specific template is assigned to trusted users to act as key recovery agents. To recover the keys, a user account must obtain the certificate. Without this certificate, even an administrator cannot retrieve the keys. In addition, it actually takes two to tango; a CA Administrator and a key recovery agent must cooperate to obtain the archived keys. One individual on their own cannot do so. As such, it is a good idea to store important certificates, such as the Key Recovery Agent, on a smart card, to ensure that it can't be duplicated and distributed.

Revocation

Certificates do a have a validity period, or time during which they may be used, and any certificate-aware application should be designed to check this time frame before approving use of the certificate. Nevertheless, keys might be compromised, and users leave the company—what then? A certificate can be revoked, and the CA periodically publishes a list, the certificate revocation list (CRL), which can be examined by the application. Windows Server 2008 added support for Online Certificate Status Protocol (OCSP) responders, which can be utilized by Windows Vista or higher. OCSP responders are more efficient at letting clients know if a certificate is valid.

An excellent analogy for CRLs is the old way in which credit cards were verified as still valid. Credit card companies used to send to stores large books that listed all credit card numbers that had been canceled. It was the responsibility of the store employee to look up a card against that list at the time of transaction to make sure it hadn't been canceled. This is exactly how a CRL works. The CRL publishing path is stamped into the certificate in the CRL Distribution Path (CDP) and is typically an HTTP path. OCSP works the way modern credit card validation works. At the time of transaction, the credit card number is sent to a service whose job is to tell the merchant if the number is still valid or not. The service returns a simple "yes" for valid or "no" for invalid; if valid, the transaction is allowed to proceed. This is how OCSP works. OCSP responders are listed in the Authority Information Access (AIA) field of the certificate.

Role Separation

Each user and administrator of certificate services plays a role. Specific CA roles are CA Administrator and CA Manager. The CA Administrator manages the CA, and the CA Manager manages certificates. The CA Administrator is not automatically granted operating system administrator privileges, and the local, domain, or enterprise admin can have her default CA administration privileges removed. Users are given Enroll rights (the right to request a certificate and, if template permissions are validated, to obtain a specific certificate). Backup Operators are given the right to back up the CA. This role separation fulfills two of the dictums of good security: provide each user with only the permissions and privileges they need to do a good job, and separate activities so that it takes two or more people to perform sensitive operations.

Note that the operating system administrators can reclaim the right to administer the CA. There is no way to permanently remove the ability of an administrator to administer a Windows system. You will have to trust the administrator to follow the security policy—and audit their activity on the computer.

Of course, membership in multiple groups can subvert this security paradigm. However, it is possible in Windows Server 2003 and later CAs to enforce role separation. If this is done, any user who has both CA Administrator and CA Manager rights will not be able to exercise either.

The only reasonable way to further protect the CA from a system administrator is to store the private keys for the CA in a Hardware Security Module (HSM) and enforce the use of multiple smart cards to access the key material. This allows you to issue *n* cards and define the need for *m* cards to be present to activate the HSM and thus activate the CA. While this can be inconvenient for rebooting CAs, it is an excellent improvement in overall security for a PKI.

Cross-Certification

Just as multiple Windows domains or forests can inadvertently multiply within an organization, so can multiple CA hierarchies be created. If this is the case and trust between the hierarchies is required or if you need to establish trust between two hierarchies belonging to different organizations, Windows Server 2003 or higher CA hierarchies can cross-certify with other Windows Server 2003 or higher CA hierarchies and some third-party product CA hierarchies. Cross-certification is obtained by issuing and exchanging cross-certification certificates between the hierarchies. Multiple constraints can be applied to limit the cross-certification.

Compliance with Standards

If you are following a specific security framework, here's how NIST, ISO 27002, and COBIT tie in to this chapter. NIST has the most specific guidance for configuring Windows, down to the level of how to configure the operating system. ISO 27002 has some higher-level guidance, and COBIT is even higher level. The relevant sections of each standard are provided.

NIST

NIST offers guidance for use of encryption in U.S. government agencies in several Special Publications:

- SP 800-133: DRAFT Recommendation for Cryptographic Key Generation
- SP 800-131A: Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths
- DRAFT SP 800-130: A Framework for Designing Cryptographic Key Management Systems
- SP 800-111: Guide to Storage Encryption Technologies for End User Devices
- SP 800-78-3: Cryptographic Algorithms and Key Sizes for Personal Identification Verification (PIV)
- SP 800-67 Rev. 1: Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher
- SP 800-56B: Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography
- SP 800-56A: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography
- SP 800-21 [Second Edition]: Guideline for Implementing Cryptography in the Federal Government

ISO 27002

ISO 27002 contains the following provisions, to which this chapter's contents are relevant.

- **12.3.1** The use of cryptographic controls should be governed by a policy that specifies the level of protection for information based on a risk assessment, and digital signatures should be used to protect the authenticity and integrity of key electronic documents.
- **12.3.2** A cryptographic key management system should be used to protect the keys associated with cryptographic controls, based on secure standards.
- **15.1.6** The use of cryptographic controls should comply with local and national laws and regulations.

COBIT

COBIT contains the following provision, to which this chapter's contents are relevant.

- **DS5.8** Policies and procedures for cryptographic key management should manage the lifecycle of keys, including their generation, use, protection, and destruction, and the keys should be properly protected.

Summary

This chapter provides a high-level overview of how encryption works, and what encryption methods are available, along with some of their relative strengths and issues. A deeper dive into encryption requires a whole book, several examples of which are included in the "References" section at the end of the chapter for those who wish to learn more.

In this chapter, we started with a brief history of encryption in order to establish a context regarding the limited lifespan of cryptographic techniques. Looking at early codes, and the progression to more modern codes, we saw how the encryption methods evolve to stay one step ahead of those who want to break the confidentiality of the protected data.

Symmetric-key cryptography evolved naturally from early methods of hiding data using mathematical transformations. In these algorithms, key exchange is a key challenge. Whoever possesses the key can decrypt the message—thus, properly secured key exchange is critical to the continued confidentiality of the data.

Public key cryptography is the next evolution of encryption. Using two keys, one public and one private, helps deal with the problem of key exchange that was encountered in symmetric-key encryption. Public Key Infrastructure (PKI) uses public key cryptography to create certificates, which are used for a variety of purposes. Finally, we looked at how to keep the certificate server protected, which is important for keeping the certificates safe.

References

Curtin, Matt. *Brute Force: Cracking the Data Encryption Standard*. Springer, 2010.

Ferguson, Niels, and Bruce Schneier. *Practical Cryptography*. Wiley, 2003.

Hershey, John. *Cryptography Demystified*. McGraw-Hill, 2002.

Katz, Jonathan, and Yehuda Lindell. *Introduction to Modern Cryptography: Principles and Protocols*. Chapman and Hall/CRC, 2007.

Paar, Christof, and Jan Pelzl. *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer, 2010.

Salomon, David. *Data Privacy and Security: Encryption and Information Hiding*. Springer, 2012.

Schneier, Bruce. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. 2nd ed. Wiley, 1996.

Schneier, Bruce. *Secrets and Lies: Digital Security in a Networked World*. Wiley, 2004.